An Integrated Fault Tolerant Robotic Controller System for High Reliability and Safety

Neville 1. Marzwell
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Kam S. 'l'so and Myron Hecht
SoHaR Incorporated
Beverly hills, CA 90211

ABSTRACT

'l'his paper describes the concepts and features of a fault-to] erant intelligent robotic control system being developed for applications that require high dependability (reliability, availability, and safety). 'l'he system consists of two major elements: a *fault tolerant con troller* and an *operator workstation*. The fault tolerant controller uses a strategy which allows for detection and recovery of hardware, operating system, and application software failures. It has a recovery time of less than 40 milliseconds, a period short enough for nearly all real time applications. Protection against higher level unsafe events (e.g., collisions) is provided by software resident in a separate operator workstation which includes features to predict collisions and reduce the human workload thereby reducing errors and enhancing safety, The fault tolerant controller can be used by itself in a wide variety of applications in industry, process control, and communications. The controller in combination with the operator workstation can be applied to robotic applications such as spaceborne extravehiclar actvities, hazardous materials handling, inspection and maintenance of high value items (e.g., space vehicles, reactor internals, or aircraft), medicine, and other tasks where a robot system failure poses a significant risk to life or property.

1. INTRODUCTION

Uncertain failure behavior and the potential for unsafe events have been significant concerns for the application of robots in some critical applications. Earlier work [1, 2, 3, 4] has defined two classes of potential hazards in robot controllers: those caused by *system level* failures, i.e., failure of the robot controller itself and those caused by *task level* failures, i.e., semantically valid commands input to the robot controller which in fact could result in collisions, unsafe movements, or other hazards.

Figure 1 shows a top level view of the fault tolerant robotics controller system based on this view. 'J'here arc two main subsystems: an operator workstation and the controller subsystem. The operator workstation is a Silicon Graphics IRIS Crimson/VGXT running the IRIX 5.2 operating system. The controller subsystem consists of a pair of redundant VME chassis running Motorola 68040 Single Board Computers under the VxWorks Release 5.] real time multitasking kernel. The operator workstation interfaces with the controller subsystem over Ethernet. The two redundant VME chassis are interfaced to each other using a high speed BIT 3 bus adapter, and each have a separate connection to the robot arm. For the purposes of test and evaluation, these systems arc being integrated with the Robotics Research 7 DOF arm located at the Jet Propulsion Laboratory (JPI.).

The approach integrates a sophisticated user interface with hardware and software fault tolerance resident in the controller. Three kcy technologies have emerged from this work:

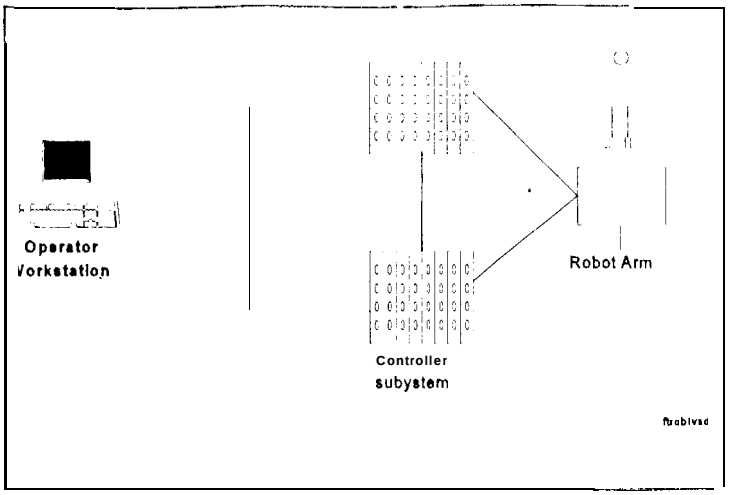

Figure 1. Top 1 .evel System View

- Comprehensive and high performance (40 msec recovery time) fault tolerance implemented as an executive layer on the VxWorks kernel.

- A robot controller implemented using I Distributed Recovery Blocks, and

- A user interface allowing for the creation of complex event sequences and providing operator feedback through a simulation of the robot arm.

The first two technologies are associated with the controller subsystem and are described in the following section. Section 3 describes some aspects of the user interface. Section 4 describes potential follow-on uses of these technologies either as separate entities or as an integrated system.

2. FAULT TOLERANT CONTROLLER

The fault tolerant controller accepts Cartesian coordinates and translates them into joint angles which are then output to servo controllers within the robot. Fault tolerance for hardware, software, and communications failures is necessary in the controller because it must respond rapidly to failures. A real-time fault-tolerant distributed architecture called the Extended Distributed Recovery Block (EDRB) [5] handles controller failures. The underlying fault tolerance algorithms and mechanisms arc based on the distributed recovery block [6] which is in turn based on the classical recovery block [7] with real time extensions. Figure 2 is a top level diagram of a robotic control system which incorporates the EDRB.

in the terminology of the EDRB, the replicated controller computers are collectively referred to as an operational node pair. One member of the node pair, called the active node, provides control and processing for the robot and sensors. The other node, referred to as the shadow, operates as a standby. The active and shadow nodes exchange frequent periodic status messages, called heartbeats, over redundant communication lines as both an indication of their states of health and

for state data updates. If the shadow node senses the absence of its companion active node's heartbeat, it will promote itself to the active status after verifying concurrence with a supervisor. The supervisor in this system is a task resident on the controller workstation. This concurrence is required in order to prevent a spurious takeover due to faulty communications in the shadow node or a false alarm due to a transient anomaly. After taking over, the newly promoted active node will induce a hardware reset and software reload of the failed node in the hope of restoring it to backup status. The supervisor itself need not be replicated because it is needed only to assist in recovery; the EDRB can function in steady state without the supervisor.

Figure 2 shows how distributed recovery blocks are implemented in the EDRB. Within both the active and shadow nodes are two versions of the task execution software, referred to as the primary and alternate routines. Under normal circumstances, the primary routine is run on the active node while the alternate routine is concurrently run on the shadow. The primary routine is coded to provide the greatest functionality, accuracy, and performance. The alternate routine provides less functionality and performance, but is coded to optimize reliability.

The EDRB tolerates a broad range of hardware, system software, and application failures including:

- Robotic task execution software not outputting a correct setpoint by the required deadline (detected by means of acceptance tests, timers, and recovered from using the alternate routine).

- Hardware or system software failures (detected by means of information encoding, timers, and recovered from by switching to the redundant processor).

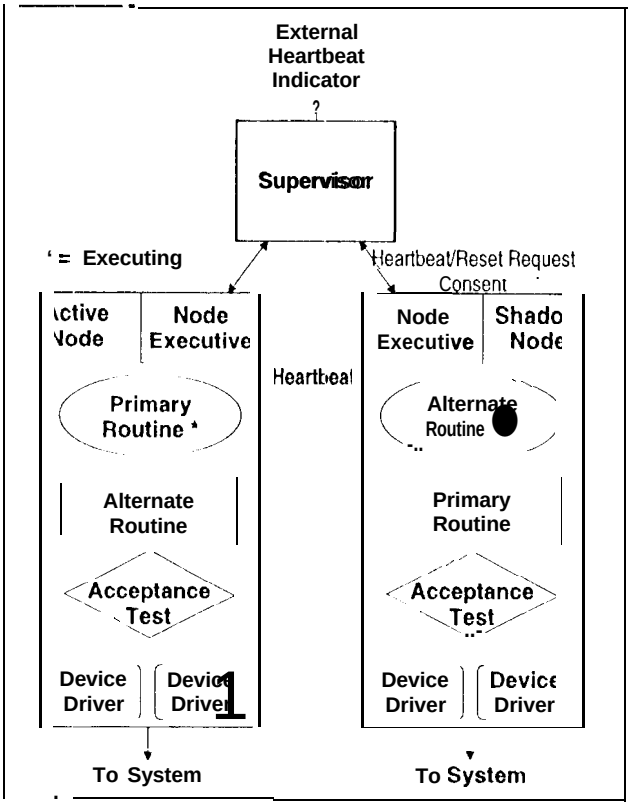

Figure 2. Software Architecture of EDRB

- Communications link failures (detected by means of encoding, and recovered from using retransmission, and redundant communication links).

- Spurious recovery actions (avoided by means of the supervisor and consideration of failure histories in the node executive).

One of the most important characteristics of the EDRB for robotic control applications is its fast response and recovery time. The algorithms used in the EDRB fault detection and recovery modules arc fast because they do not require any kind of rollback. "l'his characteristic is achieved by executing the primary and alternate routines in parallel. The EDRB provides the general framework of the primary routine, alternate routine, and acceptance test which work together to tolerate software faults. However, it is necessary to define application specific algorithms for the primary and alternate routines, as well as to define acceptance tests which dependably distinguish between correct and incorrect output.

1 'or the controller application, diversity between the the primary and alternate routines is achieved using (1) the Jacobian pseudoinverse [8] which has good tracking but cannot handle singularity, and (2) the damped least square [9] which is singularity robust but has bad tracking near singularity. The primary routine can use the Jacobian pseudoinverse to ensure good tracking, while the alternate routine, based on the damped least square, would be used when the primary fails to handle singularity. Because many of the software failures in these routines are likely to be in the mathematical operations, the alternate routine will rely on lookup tables instead of math library functions provided by the compiler.

The acceptance test is the single most critical clement of the EDRB. The two potential failure modes arc rejection of a correct result or acceptance of an incorrect result. In order to avoid these failure modes, the acceptance test must bc both simple so that it can be thoroughly verified and general so that it provides an adequate level of coverage and safety. While these are rigorous requirements, they arc feasible in robotic applications. In the free motion example, the acceptance test will determine (1) that the next setpoint is closer to the destination than the previous, (2) the difference between the observed joint angles and the command joint angles are with in an acceptable range, (3) the command joint angles arc not close to joint limits, and (4) the observed force/torque values are within an acceptable range of the gravitational force of the grasped object.

3. OPERATOR WOI{KS'1'ATION

The operator workstation provides both a user interface and additional functionality to prevent collisions and other unsafe actions. This functionality is necessary because although the fault tolerant controller can prevent unsafe actions caused by loss of controller hardware or software (i.e., *system* level hazards), it can not prevent collisions or other undesirable events which are caused by deliberate and planned motion (i.e., *task* level hazards).

The following capabilities are being developed for the operator workstation to enhance safety:

- *Graphical User Interface:* A sophisticated graphical user interface allows operators to easily command and monitor robot motions. The interface design is intended to minimize confusion and fatigue thereby prevent operator errors. Figure 3 shows an example of this interface tailored for remote surface inspect ion,

- Collision Prediction: The operator will have the ability to identify volumes of prohibited motion. Prior to performing a command, a robotic simulator running in the workstation will determine whether the motion will cause movement through prohibited areas.